

Sentiment analysis based on rhetorical structure theory: Learning deep neural networks from discourse trees

Mathias Kraus^{a,*}, Stefan Feuerriegel^a

^a*Chair for Information Systems Research, University of Freiburg, Platz der Alten Synagoge,
79098 Freiburg, Germany*

Abstract

Prominent applications of sentiment analysis are countless, including areas such as marketing, customer service and communication. The conventional bag-of-words approach for measuring sentiment merely counts term frequencies; however, it neglects the position of the terms within the discourse. As a remedy, we thus develop a discourse-aware method that builds upon the discourse structure of documents. For this purpose, we utilize rhetorical structure theory to label (sub-)clauses according to their hierarchical relationships and then assign polarity scores to individual leaves. To learn from the resulting rhetoric structure, we propose a tensor-based, tree-structured deep neural network (named RST-LSTM) in order to process the complete discourse tree. The underlying attention mechanism infers the salient passages of narrative materials. In addition, we suggest two algorithms for data augmentation (node reordering and artificial leaf insertion) that increase our training set and reduce overfitting. Our benchmarks demonstrate the superior performance of our approach. Ultimately, this work advances the status quo in natural language processing by developing algorithms that incorporate semantic information.

Keywords: Sentiment analysis, Rhetorical structure theory, Discourse tree, Tree-structured network, Long short-term memory, Attention mechanism

*Corresponding author. Mail: mathias.kraus@is.uni-freiburg.de; Tel: +49 761 203 2395; Fax: +49 761 203 2416.

Email addresses: mathias.kraus@is.uni-freiburg.de (Mathias Kraus), stefan.feuerriegel@is.uni-freiburg.de (Stefan Feuerriegel)

1. Introduction

User-generated content reveals personal opinions towards entities such as products, services or events, which can benefit organizations and businesses in improving their marketing, communication, production and procurement. In this context, sentiment analysis facilitates the extraction of subjective information from user-generated content, or narrative materials in general, by quantifying the positivity or negativity of natural language [1, 2]. Among the many applications of sentiment analysis are tracking customer opinions [3], evaluating survey responses [4], mining user reviews [5, 6, 7], trading upon financial news [8, 9] and predicting sales [6, 10].

Sentiment analysis traditionally utilizes bag-of-words approaches, which merely count the frequency of words (and tuples thereof) to obtain a mathematical representation of documents in matrix form [1, 2]. As such, these approaches are not capable of taking into consideration semantic relationships between sections and sentences of a document. Let us, for instance, consider the following two examples, which express opposite polarities: *“the movie was [good]⁺, although reviews are [bad]⁻”* and *“the movie was [bad]⁻, although reviews are [good]⁺”*. Both constitute identical representations in naïve bag-of-words models, which are unable to account for information regarding semantics and discourse.

A remedy to this problem may be found in the form of n -grams, which construct tuples of n contiguous words from a text and thus incorporate the contextual information of words into the representation of text [11]. Here research commonly utilizes n -grams of size two (bi-grams) and three (tri-grams), which are often enriched with additional features such as part-of-speech [1]. This approach allows for implicitly formalizing dependencies between adjacent words. Specific use cases are the detection of negation scopes, e.g. *“not bad”*, or compound nouns, e.g. *“computer science”*. Yet n -grams cannot encode relationships between different sections of a text. For instance, bi-grams cannot

correctly infer the sentiment of the previous example.¹

To reliably analyze the sentiment of lengthy documents, it is essential to incorporate the semantic structure. A recent review in *Science* points out that novel techniques are required to leverage semantics [12]. Hence, we propose an innovative method, based on rhetorical structure theory (RST), that incorporates the discourse structures of natural language. RST structures documents hierarchically [13] by splitting the content into (sub-)clauses called elementary discourse units (EDUs). The EDUs are then connected to form a binary discourse tree. The edges are further labeled according to the type of discourse, for instance, whether it is an elaboration or an argument. The RST tree thus helps localize essential information within documents; e. g. the concluding section of a newspaper article is typically relevant as it reports the opinion of the author. Figure 1 illustrates the discourse tree for the previous example, in which RST renders it possible to identify the intended sentiment (i. e. the subjective statement conveyed by the main clause). Hence, the goal of this work is to develop a novel approach that identifies salient passages in a document based on their position in the discourse tree and incorporates their importance in the form of weights when computing sentiment scores.

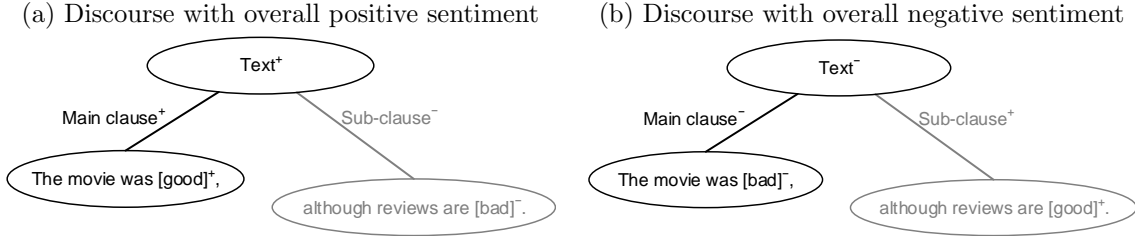


Figure 1: Illustrative examples in which the discourse tree helps identify the conveyed sentiment from the main clause.

¹Bi-grams contained in the first example are “The movie”, “movie was”, “was [good]+”, “[good]+ although”, “although reviews”, “reviews are”, “are [bad]-”. The second example includes the bi-grams “The movie”, “movie was”, “was [bad]-”, “[bad]- although”, “although reviews”, “reviews are”, “are [good]+”. Hence, the vector representations differ for each sentence, but both still suggest the same, albeit false, neutral rating.

Previous research has demonstrated that discourse-related information can successfully improve the performance of sentiment analysis (see Section 2 for details). For instance, one can reweigh the importance of passages based on their relation type [14] or depth [15] in the discourse tree. Some methods prune the discourse trees at certain thresholds to yield a tree of fixed depth, e.g. 2 or 4 levels [15]. Other approaches train machine learning classifiers based on the relation types as input features [16]. What the previous references have in common is that they try to map the tree structure onto mathematically simpler representations, since it is virtually impossible to encode unstructured data of arbitrary complexity in a fixed-length vector.

We overcome the limitations of previous works and propose a specific neural network, named RST-LSTM, for representation learning. The RST-LSTM utilizes multiple tensors to build an attention mechanism that can localize salient passages within documents by following the discourse structure. In contrast to previous works, the RST-LSTM is thus capable of exploiting the complete structure of the discourse tree. In addition, we propose two techniques for data augmentation that facilitate training and yield higher predictive accuracy.

In brief, our approach is as follows: we utilize rhetorical structure theory to represent the semantic structure of a document in the form of a hierarchical discourse tree. We then follow a dictionary-based approach to obtain sentiment scores for each leaf. The resulting tree is subsequently traversed by the RST-LSTM, thereby aggregating the sentiment scores based on the discourse structure in order to compute a sentiment score at document level. This approach thus weighs the importance of (sub-)clauses based on their position and relation in the discourse tree, which is learned during the training phase. As a consequence, this allows us to improve sentiment analysis with discourse information.

The RST-LSTM also differs from the attention mechanism in [17], which can only exploit the relation type and not the hierarchy. Furthermore, former approaches are based on traditional recursive neural networks, which are limited by the fact that they can persist information for only a few iterations [18].

Therefore, these methods struggle with complex discourses, while ours can handle even very deep tree structures.

The remainder of this paper is structured as follows. Section 2 reviews discourse parsing and RST-based sentiment analysis. To overcome the limitations of previous approaches, Section 3 introduces the RST-LSTM, as well as our algorithms for data augmentation. Section 4 describes our experimental setup, for which we evaluate the performance of our deep learning methods in comparison to common baselines (Section 5). Section 6 concludes with a summary and suggestions for future research.

2. Background

2.1. Rhetorical structure theory

Rhetorical structure theory formalizes the discourse in narrative materials by organizing sub-clauses, sentences and paragraphs into a hierarchy [13]. The premise is that a document is split into elementary discourse units, which constitute the smallest, indivisible segments. These EDUs are then connected by one of 18 different relation types, which represent edges in the discourse tree; see Table 1 for a list [19]. Each relation is further labeled by a hierarchy type, i. e. either as a nucleus (N) or a satellite (S), where a nucleus denotes a more essential unit of information, while a satellite indicates a supporting or background unit of information. We note that this hierarchy type is not necessarily an exclusive-or, since both children can be labeled as nuclei or satellites at the same time. Figure 2 presents an example of a discourse tree.

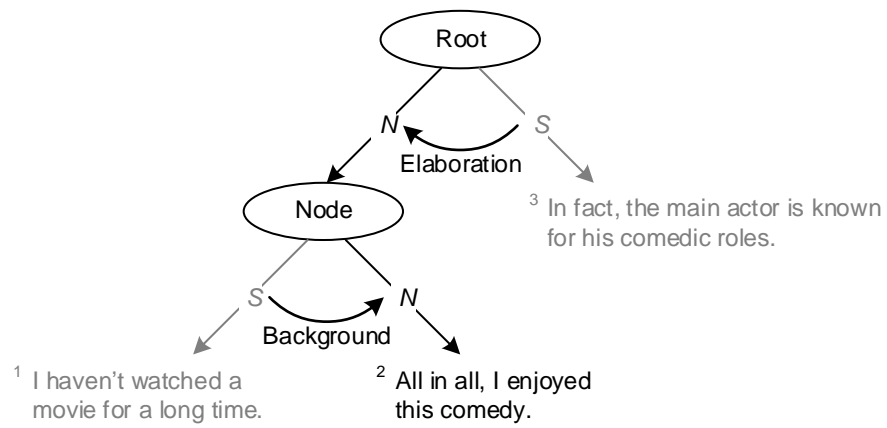


Figure 2: Example discourse tree with 3 elementary discourse units, for which N denotes a nucleus and S a satellite.

Relation type	Description
Elaboration	Satellite provides additional details about the nucleus
Joint	No specific hierarchy between EDUs
Same-unit	Links parts of one EDU to another
Background	Satellite provides information to comprehend nucleus
Attribution	Satellite contains reporting verbs or cognitive predicates for nucleus
Comparison	Refers to similarities and dissimilarities
Temporal	Describes a specific ordering in time between units
Enablement	Satellite increases the ability to perform the action in nucleus
Contrast	Describes comparability or differences
Summary	Satellite is a shorter restatement of nucleus
Condition	Realization of nucleus depends on realization of satellite
Manner-means	Satellite tends to make realization of nucleus more likely
Cause	Satellite is a reason for nucleus
Explanation	Satellite justifies information from nucleus
Evaluation	Satellite assesses nucleus
Textual-organization	Describes the composition of the document
Topic-change	Topic has changed between units
Topic-comment	One EDU annotates another

Table 1: Overview of the different relation types that connect elementary discourse units [19].

Previous research has proposed various methods for automating the parsing of discourse trees of documents. Common implementations for documents consisting of multiple paragraphs are represented by the high-level discourse analyzer HILDA [20] and the DPLP parser [21], of which the DPLP parser currently achieves the better F1-score in identifying relation types. Since this is regarded as the most challenging subtask of RST parsing, we specifically decided to utilize the DPLP parser in this work.

2.2. Sentiment analysis with RST

Previous studies have advocated different approaches for sentiment analysis that utilize the discourse tree; see Table 2.

The underlying sentiment scores of EDUs are almost exclusively calculated on the basis of pre-defined sentiment dictionaries. Common examples include

SentiWordNet [22, 19, 14, 16], hand-crafted dictionaries [23] or domain-specific dictionaries [15]. A recent approach calculates *ex ante* vector representations for the EDUs based on word embeddings [17]. This essentially resembles a high-dimensional dictionary that is, consistent with the pre-defined dictionaries, applied prior to learning the weights of the classifier. For reasons of comparability, we also utilize a dictionary-based approach in order to compute sentiment features from the content of elementary discourse units.

Among the most common methods are simple weighting rules that aggregate the sentiment scores of EDUs based on the tree structure [19, 14]. However, the weights are frequently pre-determined and hand-crafted. A different stream of research also considers hierarchy labels (nucleus or satellite) of the nodes and updates the weights based on these. Examples include approaches that focus on the top-split (i.e. the root node) of the discourse tree and scale the relative importance based on (hand-crafted) weights [23, 19, 14]. The underlying weights can also be optimized using logistic regression [24]. In contrast, other works specifically focus on the hierarchy labels at leaf level, arguing that this choice facilitates a more fine-grained evaluation [14], while neglecting the discourse tree from above. To fill the gap between top-level and leaf-level analysis, recent research also applies a recursive weighting scheme that utilizes a scaling factor to diminish the influence of increasing depth [14, 15]. Alternatively, one can prune the discourse tree at certain thresholds in order to yield a tree of fixed depth, e.g. 2 or 4 levels [15]. Some works also incorporate relation types between EDUs [19, 24, 14] or categorize them into contrastive or non-contrastive relations, which are then weighted separately [22].

The papers discussed above predominantly consider depth-related information of the RST nodes. Accordingly, they neglect the links between nodes within the tree structure. A potential remedy is to traverse the RST tree with a recursive neural network [25]; however, this approach only incorporates the edges and lacks information regarding the relation type. Closest to our work is an approach that traverses the RST tree with the help of a recursive neural network, while utilizing relation-specific composition matrices in order to build an

attention mechanism [17]. However, the recursive neural network is known to struggle with complex tree structures because of vanishing or exploding gradients and, instead, we utilize a long short-term memory. Moreover, the approach sums the representations in each recursion and, hence, cannot distinguish the hierarchy; i. e. between nucleus and satellite. Hence, the objective of this paper is to extend the previous works by advancing representation learning in order to incorporate the complete discourse tree, including relation types, tree depth and hierarchy labels.

2.3. Representation learning for sequential and tree data

Recent advances in deep neural networks have rendered it possible to learn representations of unstructured data such as sequences or text [26]. This can, for instance, be achieved by recurrent neural networks, which entail an internal architecture in the form of a directed cycle, thereby creating an internal state with which to learn dependent structures [27]. Based on these, one can process texts of arbitrary length in sequential order, while the internal state encodes the complete sequence. The underlying structure thus allows for the passage of information from one word to the next. However, in practice, information only persists for a few iterations [18]. A viable remedy is provided by the long short-term memory (LSTM) network. These enhance recurrent neural networks by capturing long dependencies among input signals [28]. We thus utilize LSTMs as part of our baselines later, as well as inside all tree-structured networks.

Previous research has proposed a Tree-LSTM that can deal with representation learning, for instance for trees from RST, and we thus rely upon it as another baseline. This tree-structured LSTM network traverses trees bottom-up in order to generate representations of the underlying structure [29]. The Tree-LSTM computes a representation for each parent node based on its immediate children and does so recursively until the root of the tree is reached. It thereby stacks individual LSTMs such that they reflect the tree structure from the input. However, the Tree-LSTM provides no possibility of incorporating additional information from the discourse trees, such as the relation type. We

Reference	Datasets	Predicted variable	Weight optimizations	EDU features	RST parser	Model	Considered RST features		
							Relation type	Tree depth	Nucleus/satellite
Bhatia et al. [25]	Rotten tomatoes reviews, IMDb reviews	User rating	Handcrafted weighting factor or classifier-based	Dictionary-based sentiment score	DPLP	Hierarchical weighting rule Recursively traversal of RST tree with recurrent neural network	✓ ✓	✓ ✓	✓ ✓
Chenlo et al. [24]	BLOGS06	Blogpost ranking	Weights optimized by logistic regression	Dictionary-based sentiment score	SPADE	Top-split weighting	✓	✓	✓
Heerschop et al. [19]	Rotten tomatoes reviews	User rating	Handcrafted weighting factor or optimized by genetic algorithm	Dictionary-based sentiment score	SPADE	Position-based weighting rule Top-split weighting rule Relation-type weighting	✓ ✓ ✓	✓ Up to level 1 ✓	✓ ✓ ✓
Hogenboom et al. [14]	Rotten tomatoes reviews	User rating	Handcrafted weighting factor or particle swarm optimization	Dictionary-based sentiment score	HILDA SPADE	Position-based weighting Top-split weighting Bottom-split weighting Hierarchical weighting	✓ ✓ ✓ ✓	✓ Up to level 1 ✓ ✓	✓ ✓ ✓ ✓
Hogenboom et al. [16]	Rotten tomatoes reviews, Multi-domain collection	User rating	Weights optimized by SVM	Dictionary-based sentiment score	SPADE	Top- and leaf-split weighting of 14 relation types	✓	✓	✓
Ji and Smith [17]	Yelp reviews, Rotten tomatoes reviews, Congressional debates, Congressional bill corpus, Media frames corpus	User rating or class labels	Transformation of RST tree into dependency structure	Hidden states of bidirectional LSTM on word embeddings	DPLP	Average of all EDUs Select root node of dependency RST tree Tree-structured neural network with attention-weights	✓ ✓ ✓ ✓	✓ Up to level 1 ✓ ✓	✓ ✓ ✓ ✓
Märkle-Huß et al. [15]	Financial disclosures	Directional stock price change	Weights optimized by grid-search	Dictionary-based sentiment score	HILDA	Hierarchical weighting rule Random forest on pruned tree	✓ ✓	✓ Up to level 2 or 4	✓ ✓
Taboada et al. [23]	Opinions reviews, Rotten tomatoes reviews	User rating	Handcrafted weights	Dictionary-based sentiment score	SPADE	Top-split weighting	✓	Up to level 1	✓
Zirn et al. [22]	MDSA	User rating		Dictionary-based sentiment score		Markov logic network	Contrastive and non-contrastive	✓	✓
This paper	Rotten tomatoes reviews, IMDb reviews	User rating	Additional data augmentation	Dictionary-based sentiment score	DPLP	RST-LSTM	✓	✓	✓

Table 2: Comparison of methods for sentiment analysis utilizing the discourse structure.

thus later extend the naïve Tree-LSTM through two additional tensor-based attention mechanisms, resulting in an RST-LSTM that allows us to utilize the complete set of information encoded in discourse tree.

3. Discourse-based sentiment analysis with deep learning

This section introduces our discourse-based methodology, which infers sentiment scores from textual materials. Figure 3 illustrates the underlying framework and divides the procedure into steps for discourse parsing, computing low-level polarity features, data augmentation and prediction. The prediction phase implements either one of the baselines (e.g. LSTM or Tree-LSTM) or our proposed RST-LSTM.

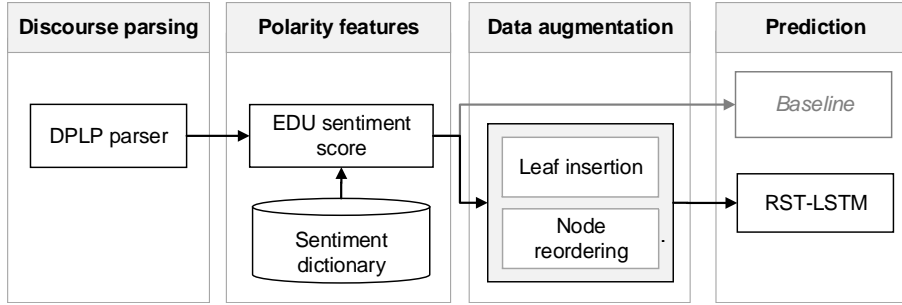


Figure 3: Research framework evaluating the gains in predictive performance from combining our RST-LSTM and data augmentation in comparison to the baselines.

3.1. Discourse parsing

We generate discourse trees for our datasets by utilizing the DPLP parser [21]. For sake of simplicity, we introduce the following notation. We denote the relation type of node i as $\rho_i \in \{\text{elaboration, argument, } \dots\}$. The complete list of relation types is given in Table 1. Furthermore, let π_i specify a path from the root to a certain node i in the tree given by a continuous list of nucleus (N) or satellite (S) traversals. We then denote the relation type of i as ρ_{π_i} . For example, ρ_{SN} refers to the relation type of node SN , which is the nucleus

in the first satellite branch. In this regard, the symbol R represents the root node of a discourse tree and, hence, ρ_R provides the relation type of the root. Furthermore, we denote the list with the relation types of all nodes in an RST tree as ρ_{RST} . We also introduce $\tau_i \in \{\text{nucleus}, \text{satellite}\}$ as the hierarchy type of node i . Additionally, we specify the list of all hierarchy types in an RST tree as τ_{RST} .

3.2. Polarity features

We follow common procedures in sentiment analysis and utilize a pre-defined dictionary that labels terms as positive or negative [1, 2]. This approach has multiple advantages, as it is domain-independent and works reliably even with few training observations. In addition, one can easily exchange the underlying dictionary for one that not only measures polarity or negativity, but is concerned with other language concepts such as subjectivity, certainty or the domain-specific tone. Our experimental results are based on the SentiWordNet 3.0 dictionary [30], which provides sentiment labels for 117,659 words.

Based on the sentiment labels at word level, we then proceed to compute a sentiment score σ_i for each EDU i via

$$\sigma_i = \frac{1}{|\{w \mid w \in i\}|} \sum_{w \in i} \text{pos}(w) - \text{neg}(w), \quad (1)$$

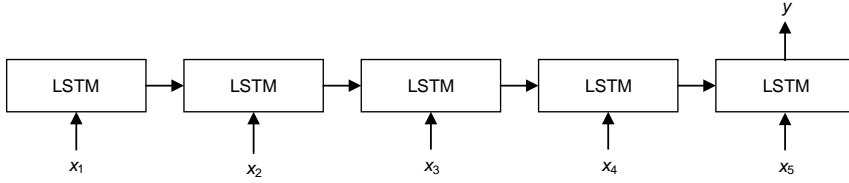
where we iterate over the words w in EDU i , while $\text{pos}(w)$ and $\text{neg}(w)$ are the positivity and negativity scores for word w according to SentiWordNet. The resulting sentiment value σ_i thus represent the low-level features that later serve as input to our predictive models.

3.3. Tree-LSTM baseline

We draw upon the Tree-LSTM as a baseline, since it is widely regarded as the status quo for tree learning [29]. The Tree-LSTM takes a discourse tree as input and then processes EDU features while incorporating their position in the tree. For this purpose, it stacks individual LSTMs in the form of that tree (cf.

Figure 4) and adapts the ideas of both a memory cell and gates from traditional LSTMs, but extends these concepts to tree structures [29]. Here the underlying LSTM helps to overcome the problem of exploding gradients.

(a) Sequential LSTM chain



(b) Tree-LSTM structure

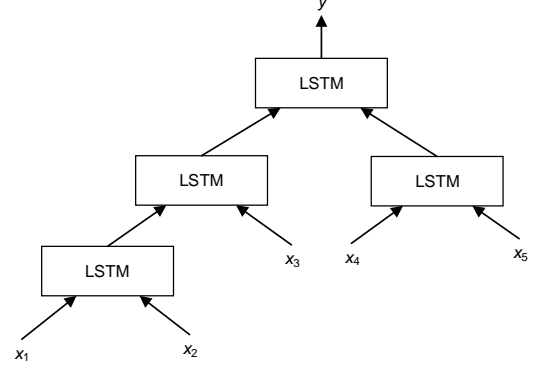


Figure 4: Schematic illustrations that show the composition of LSTM chains (left) and Tree-LSTM networks (right) for input $[x_1, x_2, x_3, x_4, x_5]$ and output y . The LSTM network processes the input sequentially, while the Tree-LSTM network traverses the tree structure.

In the Tree-LSTM, each node j from the discourse tree is translated into a single LSTM unit, which comprises an input gate i_j , an output gate o_j , a memory cell c_j and hidden state h_j . In contrast to the standard LSTM, the Tree-LSTM contains not a single forget gate, but one forget gate f_{jk} for each child k . This allows each parent node to recursively compute a representation from its immediate children. The input vectors to each LSTM unit are given by the hidden state h_k and the memory cell c_k from each child $k \in C(j)$, where $C(j)$ is the set of children of parent j . This layout of arranging connections renders it possible for the Tree-LSTM to pass information upward in the tree, since every node can incorporate selected information from each child-LSTM. Figure 5 details the connection between the gates in a Tree-LSTM.

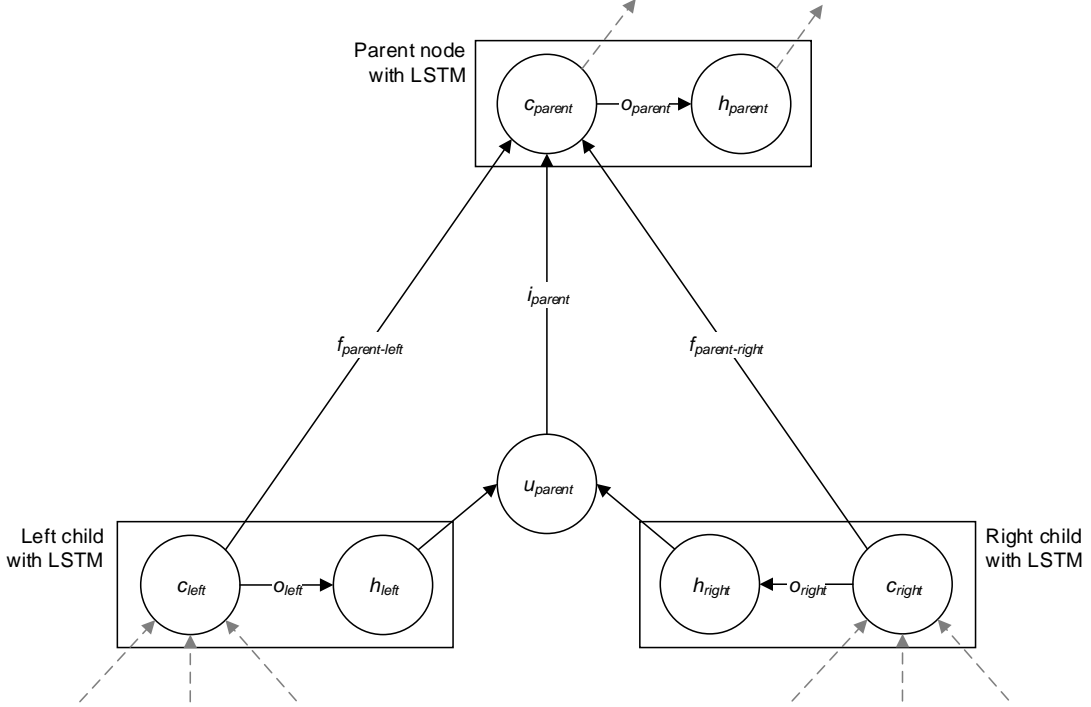


Figure 5: This schematic illustration shows the composition of the memory state c_{parent} and the hidden state h_{parent} in a Tree-LSTM with two child nodes.

Our experiments later compare the performance of two different architectures of Tree-LSTM models, namely, the child-sum and N -ary Tree-LSTM [29]. Both are common in research, but vary in their connections between input and output gates. The former, the child-sum Tree-LSTM, sums the hidden states h_k from the children $k \in C(j)$ in order to obtain a single input to the hidden state $\tilde{h}_{\text{parent}}$ of the parent. This approach loses any information regarding the order of the children, since it uses the same weights $U^{(i)}$, $U^{(f)}$, $U^{(o)}$ and $U^{(u)}$ for all children. In contrast, the N -ary Tree-LSTM requires a fixed, pre-defined number of $N = |C(j)|$ children for each inner node. It then combines the child nodes by weighting their hidden states based on parameters $U_m^{(i)}$, $U_{km}^{(f)}$, $U_m^{(o)}$ and $U_m^{(u)}$ dependent on the index $m = 1, \dots, N$ of the child.

Mathematically, the child-sum Tree-LSTM transition equations are defined as

$$\tilde{h}_j = \sum_{k \in C(j)} h_k, \quad (2)$$

$$i_j = \text{sigmoid} \left(W^{(i)} x_j + U^{(i)} \tilde{h}_j + b^{(i)} \right), \quad (3)$$

$$f_{jk} = \text{sigmoid} \left(W^{(f)} x_j + U^{(f)} h_k + b^{(f)} \right), \quad (4)$$

$$o_j = \text{sigmoid} \left(W^{(o)} x_j + U^{(o)} \tilde{h}_j + b^{(o)} \right), \quad (5)$$

$$u_j = \tanh \left(W^{(u)} x_j + U^{(u)} \tilde{h}_j + b^{(u)} \right), \quad (6)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot c_k, \quad (7)$$

$$h_j = o_j \odot \tanh(c_j), \quad (8)$$

where \odot denotes the element-wise multiplication. Moreover, the above equations contain the weights $W^{(i)}$, $W^{(f)}$, $W^{(o)}$, $W^{(u)}$ and $b^{(i)}$, $b^{(f)}$, $b^{(o)}$, $b^{(u)}$ that must be learned from the data. Similarly, the N -ary Tree-LSTM obtains its memory cell and hidden state via

$$i_j = \text{sigmoid} \left(W^{(i)} x_j + \sum_{m=1}^N U_m^{(i)} h_{jm} + b^{(i)} \right), \quad (9)$$

$$f_{jk} = \text{sigmoid} \left(W^{(f)} x_j + \sum_{m=1}^N U_{km}^{(f)} h_{jm} + b^{(f)} \right), \quad (10)$$

$$o_j = \text{sigmoid} \left(W^{(o)} x_j + \sum_{m=1}^N U_m^{(o)} h_{jm} + b^{(o)} \right), \quad (11)$$

$$u_j = \tanh \left(W^{(u)} x_j + \sum_{m=1}^N U_m^{(u)} h_{jm} + b^{(u)} \right), \quad (12)$$

$$c_j = i_j \odot u_j + \sum_{m=1}^N f_{jm} \odot c_{jm}, \quad (13)$$

$$h_j = o_j \odot \tanh(c_j). \quad (14)$$

In order to make sentiment predictions from the Tree-LSTM at the root

node, we introduce an additional feedforward classification layer. Here we utilize a softmax classifier that predicts a class label y from the hidden state h_{root} of the root node. The softmax layer entails further weights $W^{(s)}$ and $b^{(s)}$, based on which it computes the probability $p(\hat{y} | h_{\text{root}})$ of the tree belonging to class \hat{y} via

$$y = \arg \max_{\hat{y}} p(\hat{y} | h_{\text{root}}) = \arg \max_{\hat{y}} \text{softmax} \left(W^{(s)} h_{\text{root}} + b^{(s)} \right), \quad (15)$$

with the negative log-likelihood of the true class label y as the cost function [26].

3.4. RST-LSTM

This section extends the previous Tree-LSTMs through an additional attention mechanism. As a result, this yields two variants of tensor-based Tree-LSTMs, which we refer to as RST-LSTM. The RST-LSTM introduces two modifications that allow us to incorporate (1) the relation type between two nodes and (2) the hierarchy type (i.e. nucleus or satellite).

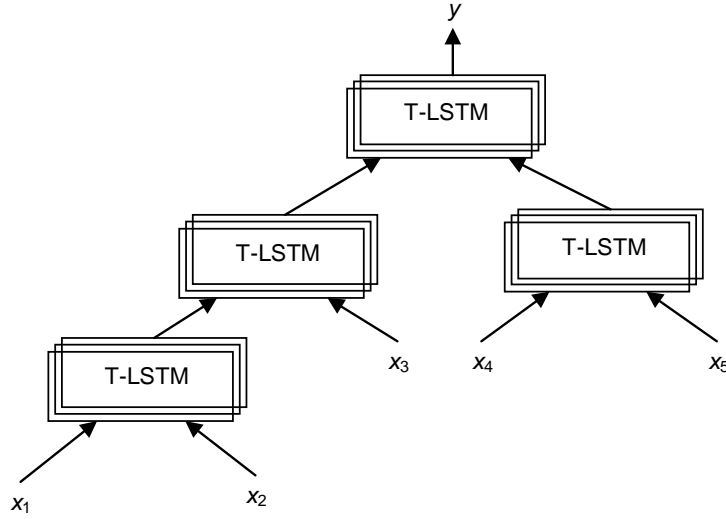


Figure 6: Schematic illustration showing the tensor idea of the RST-LSTM. In contrast to the traditional Tree-LSTM, the RST-LSTM stacks multiple LSTM units to incorporate the relation type between two nodes.

In order to include the relation type, we replace the global LSTM that serves all nodes with one that is dependent on the relation type $r \in \{1, \dots, n\}$. Figure 6 visualizes the idea schematically. We then select an LSTM_{ρ_i} for each node depending on its relation type ρ_i .

We incorporate the hierarchy type τ_i (i.e. nucleus or satellite) by additionally weighting the cell state c_j and the hidden state h_j before they enter the above tensor-based LSTM. For this purpose, we introduce tensor-based weights

$$\mathbf{W}^{(c)} = \begin{bmatrix} W_{\text{nucleus}}^{(c)}; W_{\text{satellite}}^{(c)} \end{bmatrix}, \quad (16)$$

$$\mathbf{W}^{(h)} = \begin{bmatrix} W_{\text{nucleus}}^{(h)}; W_{\text{satellite}}^{(h)} \end{bmatrix}. \quad (17)$$

We then choose the weights according to the hierarchy type τ_i in the tree. This allows us to additionally discriminate the influence of nuclei and satellites.

Accordingly, the RST-LSTM must simultaneously optimize both the tensor-based LSTM, as well as the hierarchy-related tensors $\mathbf{W}^{(c)}$ and $\mathbf{W}^{(h)}$ based on a combined objective function. We thus rearrange them as rank-3 tensors as follows: let $\mathbf{W}^{(x)}[r, :]$ indicate the weight tensor for relation type r and $\mathbf{W}^{(x)}[l, :]$ denote the weight tensor for a hierarchy type $l \in \{\text{nucleus}, \text{satellite}\}$. On this basis, we now specify the new, updated equations for calculating the cell and

hidden state. As such, the child-sum RST-LSTM computes

$$\hat{h}_j = \mathbf{W}^{(h)}[l, :] h_j, \quad (18)$$

$$\hat{c}_j = \mathbf{W}^{(c)}[l, :] c_j, \quad (19)$$

$$\tilde{h}_j = \sum_{k \in C(j)} \hat{h}_k, \quad (20)$$

$$i_j = \text{sigmoid} \left(\mathbf{W}^{(i)} x_j + \mathbf{U}^{(i)}[r, :] \tilde{h}_j + \mathbf{b}^{(i)}[r, :] \right), \quad (21)$$

$$f_{jk} = \text{sigmoid} \left(\mathbf{W}^{(f)} x_j + \mathbf{U}^{(f)}[r, :] \hat{h}_k + \mathbf{b}^{(f)}[r, :] \right), \quad (22)$$

$$o_j = \text{sigmoid} \left(\mathbf{W}^{(o)} x_j + \mathbf{U}^{(o)}[r, :] \tilde{h}_j + \mathbf{b}^{(o)}[r, :] \right), \quad (23)$$

$$u_j = \tanh \left(\mathbf{W}^{(u)} x_j + \mathbf{U}^{(u)}[r, :] \tilde{h}_j + \mathbf{b}^{(u)}[r, :] \right), \quad (24)$$

$$c_j = i_j \odot u_j + \sum_{k \in C(j)} f_{jk} \odot \hat{c}_k, \quad (25)$$

$$h_j = o_j \odot \tanh(c_j). \quad (26)$$

Similarly, the N -ary RST-LSTM computes its representations via

$$\hat{h}_j = \mathbf{W}^{(h)}[l, :] h_j, \quad (27)$$

$$\hat{c}_j = \mathbf{W}^{(c)}[l, :] c_j, \quad (28)$$

$$i_j = \text{sigmoid} \left(\mathbf{W}^{(i)} x_j + \sum_{m=1}^N \mathbf{U}_m^{(i)}[r, :] \hat{h}_{jm} + \mathbf{b}^{(i)}[r, :] \right), \quad (29)$$

$$f_{jk} = \text{sigmoid} \left(\mathbf{W}^{(f)} x_j + \sum_{m=1}^N \mathbf{U}_{km}^{(f)}[r, :] \hat{h}_{jm} + \mathbf{b}^{(f)}[r, :] \right), \quad (30)$$

$$o_j = \text{sigmoid} \left(\mathbf{W}^{(o)} x_j + \sum_{m=1}^N \mathbf{U}_m^{(o)}[r, :] \hat{h}_{jm} + \mathbf{b}^{(o)}[r, :] \right), \quad (31)$$

$$u_j = \tanh \left(\mathbf{W}^{(u)} x_j + \sum_{m=1}^N \mathbf{U}_m^{(u)}[r, :] \hat{h}_{jm} + \mathbf{b}^{(u)}[r, :] \right), \quad (32)$$

$$c_j = i_j \odot u_j + \sum_{m=1}^N f_{jm} \odot \hat{c}_{jm}, \quad (33)$$

$$h_j = o_j \odot \tanh(c_j). \quad (34)$$

As a result, both the N -ary and child-sum RST-LSTM integrate the complete discourse tree into the neural network. As opposed to the works in the literature review, this approach allows us to encode both the relation type *and* the hierarchy type.

3.5. Training data augmentation

Deep neural networks typically feature a complex structure with thousands of weights that need to be trained, which makes them prone to overfitting. A viable remedy is to artificially increase the number of training samples in order to better tune parameters. Such approaches are common in computer vision, where one extracts different crops from the same image and later considers each as a training instance. We thus propose similar techniques for tree structures that enlarge our training set. These algorithms take a tree as input and then slightly modify its structure in each epoch of training (one full training cycle on the training set). The first variant, called node reordering, swaps sub-trees, while the second, artificial leaf insertion, randomly exchanges a leaf for a node with two new children.

3.5.1. Node reordering

Node reordering rearranges inner nodes while trying to preserve the inherent structure. That is, the text passages inside the nodes must keep their original order since the content might otherwise change its meaning or grammatical structure. Our approach thus randomly chooses an inner node n and relocates it to the position of its sibling m in the tree. The sibling m is then moved down the tree and becomes a child of n . Afterwards, the previous position of n is filled by one of its former children. As a result, the order of l , r and m from left to right is unchanged. The corresponding algorithm for an inner node n is sketched in Figure 7.

Altogether, this approach for data augmentation tries to modify the structure slightly, thereby generating potentially different representations of the same tree. The extent of reordering depends on the level of n , since a reordering of

a node at a higher level usually has a larger effect on the overall tree structure compared to a node at a lower level.

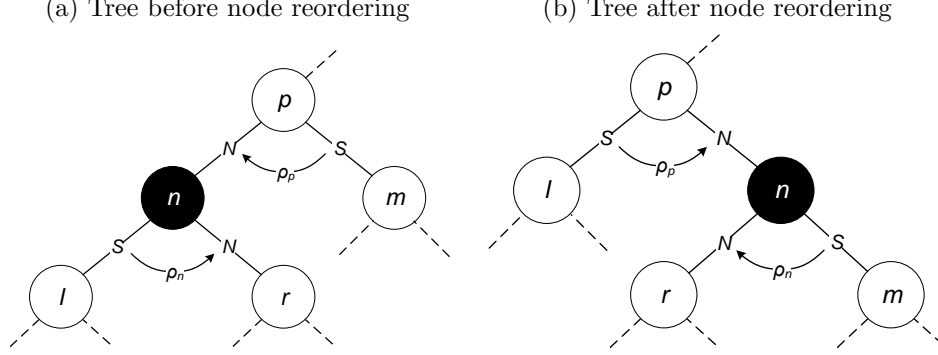


Figure 7: Schematic illustration of node reordering taking place at node n . The original tree structure is on the left, while the right shows the tree structure after node reordering.

3.5.2. Artificial leaf insertion

Artificial leaf insertion allows us to grow larger trees and helps our method to better learn complex trees. The insertion of leaves into a sub-tree is depicted in Figure 8. This approach randomly picks a leaf n from the tree and appends two newly created child nodes l and r which, subsequently present the leaves, while the n becomes an inner node. We compute σ_l and σ_r by multiplying σ_n by random weights $\omega \in [0, 1]$ and $(1 - \omega)$, i. e.

$$\sigma_l = \omega \odot \sigma_n, \quad (35)$$

$$\sigma_r = (1 - \omega) \odot \sigma_n. \quad (36)$$

These update rules thus try to keep the overall information unchanged, but distribute the values from n into two separate children given a certain ratio ω . We finally choose the relation type ρ_n and the hierarchy type τ_n randomly.

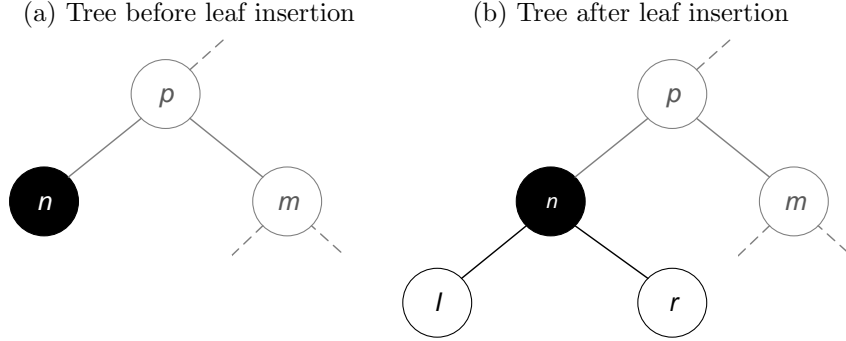


Figure 8: Schematic visualization of artificial leaf insertion at node n . The original tree structure is displayed on the left, while the right illustrates the tree structure after artificial leaf insertion.

4. Experimental setup

4.1. Datasets

We build upon earlier work and utilize two common datasets comprising movie reviews and corresponding user ratings. The first consists of 2000 movie reviews from Rotten Tomatoes [31], for which we perform 10-fold cross-validation and then average the predictive performance across splits. The second dataset comprises 50,000 reviews from the Internet Movie Database (IMDb), which are split evenly into 25,000 reviews for training and 25,000 for testing [32]. It includes, at most, 30 reviews for any one movie, since reviews for the same movie tend to have correlated ratings. Furthermore, the training and test sets contain a disjoint set of movies to avoid correlation based on movie-specific terms. Within the training and test sets, 12,500 reviews are labeled as positive and 12,500 reviews as negative in order to provide a balanced sample.

All reviews are preprocessed as follows: we perform tokenization, convert all characters to lowercase, and conduct stemming. The latter maps inflected words onto a base form, e.g. “*enjoyed*” and “*enjoying*” are both reduced to “*enjoy*” [33].

4.2. Descriptive statistics

In total, the Rotten Tomatoes corpus contains 15,462 positive, 20,153 negative and 842,537 neutral terms as defined by the SentiWordNet dictionary. Hence, 1.76 % of the words are labeled positive, while 2.29 % convey a negative connotation. The IMDb reviews include 213,723 positive, 255,335 negative and 9,031,977 neutral words, resulting in 2.25 % of the words being positive and 2.69 % being negative.

The resulting discourse trees exhibit the following characteristics. In the case of reviews from Rotten Tomatoes, they entail 51.09 EDUs on average, while this number plummets to 19.79 EDUs for IMDb reviews. The largest discourse tree contains 154 levels. Table 3 reports the relation types and corresponding frequencies in the corpus.

Relation	Dataset 1: Rotten Tomatoes reviews		Dataset 2: IMDb reviews	
	Count	Percentage	Count	Percentage
Elaboration	111,330	48.24 %	677,330	53.21 %
Joint	58,587	25.39 %	126,524	9.94 %
Attribution	13,048	5.65 %	74,119	5.82 %
Textual-organization	12,788	5.54 %	235,760	18.52 %
Same-unit	12,572	5.45 %	106,530	8.37 %
Topic-change	6950	3.01 %	3219	0.25 %
Contrast	4404	1.91 %	7872	0.62 %
Cause	2344	1.02 %	5467	0.43 %
Explanation	2214	0.96 %	4070	0.32 %
Condition	1951	0.85 %	19,949	1.57 %
Manner-means	1632	0.71 %	2618	0.21 %
Temporal	1382	0.60 %	5695	0.45 %
Comparison	854	0.37 %	3491	0.27 %
Background	346	0.15 %	5	0.00 %
Topic-comment	264	0.11 %	189	0.01 %
Summary	80	0.03 %	50	0.00 %
Evaluation	22	0.01 %	14	0.00 %
Enablement	0	0.00 %	3	0.00 %
Total	230,768	100.00 %	1,272,905	100.00 %

Table 3: Descriptive statistics of different relation types in our datasets.

4.3. Bag-of-words baseline

We construct naïve benchmarks with bag-of-words as follows. We count term frequencies (tf) and convert the numerical features into a document-term matrix. As a second baseline, we also scale the term frequencies using the term frequency-inverse document frequency approach (tf-idf), which puts stronger weights on characteristic terms [11]. Both feature spaces are then inserted a random forest, since this traditional machine learning classifier can detect highly non-linear relationships but still yields a satisfactory performance out-of-the-box. These benchmarks allow us to distinguish the sentiment conveyed by words from that conveyed by the discourse structure.

4.4. Model evaluation

We proceed as follows in order to tune the model parameters (see Table 4). In the case of the random forest baseline, we identify the optimal parameters utilizing a grid search together with 10-fold cross-validation applied to the training set. In contrast, we reduce the computational requirements of the deep learning architectures by taking 20 % of the training data as a validation set in each epoch. After each epoch, we shuffle the observations and enlarge our training set by constructing additional samples based on our technique for data augmentation.

Predictive model	Parameter	Tuning range
Random forest	Number of randomly-sampled variables	1, 2, 3, 5, 7
	Number of trees	100, 200, 500, 1000
	Maximum depth of trees	1, 5, 10, 20, 50, 100, Unconstrained
LSTM	Learning rate	0.0001, 0.001
	Regularization strength	0.0001, 0.001, 0.01
Tree-LSTM	Learning rate	0.00001, 0.0001, 0.001
	Regularization strength	0.001, 0.01, 0.05
RST-LSTM	Learning rate	0.00001, 0.0001, 0.001
	Regularization strength	0.001, 0.01, 0.05

Table 4: Overview of model parameters and their tuning ranges.

5. Results

In this section, we evaluate the performance of our RST-LSTM and compare it to the previous baselines. The evaluation provides evidence that incorporating semantic structure into the task of sentiment analysis improves the predictive performance.

5.1. Dataset 1: movie reviews from Rotten Tomatoes

Table 5 details the prediction results for the dataset featuring movie reviews from Rotten Tomatoes. The naïve benchmark with tf-idf features yields

a balanced accuracy of 0.746 and an F1-score of 0.763. The approach with tf-features, as well as the LSTM, achieves a similar performance. Here we see no clear indication that one of the baselines is consistently superior to another.

When comparing the pruned RST trees, we observe that a higher pruning depth also increases the predictive performance. For instance, the best results stem from an RST tree pruned at level 4, thereby obtaining a balanced accuracy of 0.758 and an F1-score of 0.753. However, this still ranges below the best-performing naïve benchmarks. Moreover, the inclusion of the relation type ρ_R diminishes the predictive power compared to the approach with only sentiment scores $[\sigma_N, \sigma_S]$, amounting to a decrease of 0.007 in the balanced accuracy and 0.006 in the F1-score.

The simple tree learning based on the Tree-LSTM outperforms all of the previous benchmarks. It achieves a balanced accuracy of up to 0.774 and an F1-score of 0.781. Nevertheless, the Tree-LSTM is further surpassed by the RST-LSTM, which achieves an equal balanced accuracy of 0.777, but boosts the F1-score to 0.796. This amounts to an additional improvement of 0.033 (i.e. 4.2%) in the F1-score. Altogether, the RST-LSTM benefits from the discourse-related information and thus performs best overall.

Finally, we additionally note the following patterns: there is no consistent indication that either the child-sum or N -ary variant is consistently superior. However, the N -ary models usually benefit more strongly from data augmentation due to their additional degrees of freedom. By comparing the underlying algorithms for data augmentation, the results indicate a greater increase in predictive power from leaf insertion as compared to node reordering. Yet the best performing models generally facilitate both techniques, which is not surprising given that the dataset consists of only 2000 samples. The RST-based approaches also outperform models utilizing actual words as features. This suggests that a large portion of sentiment-related information is encoded in the discourse structure.

Method	Variant	Features	Data augmentation	Accuracy	Balanced accuracy	Sensitivity	Specificity	F1-score
NAIVE BENCHMARK WITHOUT RST								
Random forest		[tf]	-	0.761	0.762	0.798	0.726	0.752
Random forest		[tf-idf]	-	0.746	0.746	0.671	0.821	0.763
DEEP LEARNING WITHOUT RST								
LSTM		$[\sigma_{\text{word}}]$	-	0.758	0.759	0.735	0.765	0.758
BENCHMARKS WITH PRUNED RST TREE								
Random forest		$[\sigma_R]$	-	0.732	0.732	0.743	0.722	0.729
Random forest		$[\sigma_N, \sigma_S]$	-	0.754	0.754	0.760	0.749	0.752
Random forest		$[\sigma_N, \sigma_S, \sigma_{NN}, \sigma_{NS}, \sigma_{SN}, \sigma_{SS}]$	-	0.758	0.758	0.778	0.739	0.753
Random forest		$[\sigma_N, \dots, \sigma_{SSS}]$	-	0.763	0.764	0.772	0.755	0.761
Random forest		$[\sigma_N, \sigma_S, \rho_R]$	-	0.747	0.747	0.749	0.746	0.746
SIMPLE TREE LEARNING								
Tree-LSTM	Child-sum	$[\sigma_{\text{EDU}}]$	-	0.774	0.774	0.740	0.807	0.781
Tree-LSTM	N-ary	$[\sigma_{\text{EDU}}]$	-	0.769	0.769	0.730	0.808	0.778
RST-BASED TREE LEARNING								
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	-	0.775	0.775	0.734	0.815	0.784
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering	0.764	0.764	0.749	0.789	0.764
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Leaf insertion	0.775	0.774	0.740	0.809	0.782
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering & leaf insertion	0.774	0.773	0.721	0.825	0.785
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	-	0.767	0.767	0.735	0.798	0.775
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering	0.774	0.774	0.746	0.802	0.780
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Leaf insertion	0.773	0.773	0.734	0.812	0.782
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering & leaf insertion	0.777	0.777	0.734	0.820	0.796

Table 5: Predictive performance reported for the test set from the small-scale dataset with 2000 movie reviews from Rotten Tomatoes. The best values for each metric are highlighted in bold.

5.2. Dataset 2: IMDb movie reviews

Table 6 reports the predictive results for the larger of the two datasets, which is based on 50,000 IMDb movie reviews. The random forest with tf-idf achieves a performance superior to the previous task, yielding an accuracy of 0.825 and an F1-score of 0.823. With regard to deep learning, the LSTM baseline improves the accuracy by 0.006 to 0.831, but shrinks the F1-score slightly by 0.002.

The accuracy is further increased by utilizing pruned discourse trees. Here the best-performing feature set is again obtained by cutting the tree at level 4, yielding an accuracy of 0.839 (an increase of 0.014 over the naïve benchmarks) and an F1-score of 0.837 (an improvement of 0.006). We also observe mixed results in terms of performance when incorporating the relation type ρ_R .

Again, tree-structured LSTMs outperform all previous baselines. For instance, the N -ary Tree-LSTM raises both the accuracy and the F1-score of the naïve baselines by 0.024. This results in an accuracy of 0.849 (i.e. 2.9% compared to the Tree-LSTM) and an F1-score of 0.847 (i.e. also 2.9%).

Our RST-LSTMs achieve the best performance overall, with an accuracy of 0.850 and an F1-score of 0.849, by utilizing data augmentation. However, we again find no general pattern indicating that one technique for enlarging the training set scores better than the other. A potential reason for the smaller improvements obtained from data augmentation compared to the previous dataset might be the larger number of training samples.

5.3. Comparison

We additionally compare our RST-LSTM to the relation-specific attention mechanism in [17], which, in contrast, sums the representations in each recursive call and hence cannot distinguish between nucleus and satellite. In addition, their approach utilizes a recursive neural network, which is known to suffer from vanishing or exploding gradients [18]. In response to such shortcomings, we decided to draw upon a long short-term memory.

We proceed as follows in order to specifically evaluate the attention mechanism itself and leave all other parameters unchanged (i.e. identical to the

Method	Variant	Features	Data augmentation	Accuracy	Balanced accuracy	Sensitivity	Specificity	F1-score
NAIVE BENCHMARK WITHOUT RST								
Random forest		[tf]	-	0.803	0.803	0.792	0.814	0.801
Random forest		[tf-idf]	-	0.825	0.825	0.835	0.815	0.823
DEEP LEARNING WITHOUT RST								
LSTM		$[\sigma_{\text{word}}]$	-	0.831	0.831	0.889	0.773	0.821
BENCHMARKS WITH PRUNED RST TREE								
Random forest		$[\sigma_R]$	-	0.806	0.806	0.808	0.803	0.805
Random forest		$[\sigma_N, \sigma_S]$	-	0.833	0.833	0.843	0.824	0.832
Random forest		$[\sigma_N, \sigma_S, \sigma_{NN}, \sigma_{NS}, \sigma_{SN}, \sigma_{SS}]$	-	0.837	0.837	0.849	0.825	0.835
Random forest		$[\sigma_N, \dots, \sigma_{SSS}]$	-	0.839	0.839	0.854	0.824	0.837
Random forest		$[\sigma_N, \sigma_S, \rho_R]$	-	0.834	0.834	0.843	0.825	0.832
SIMPLE TREE LEARNING								
Tree-LSTM	Child-sum	$[\sigma_{\text{EDU}}]$	-	0.845	0.845	0.831	0.858	0.847
Tree-LSTM	N-ary	$[\sigma_{\text{EDU}}]$	-	0.849	0.849	0.857	0.840	0.847
RST-BASED TREE LEARNING								
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	-	0.850	0.850	0.863	0.837	0.848
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering	0.850	0.850	0.882	0.818	0.845
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Leaf insertion	0.849	0.849	0.852	0.846	0.849
RST-LSTM	Child-sum	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering & leaf insertion	0.850	0.850	0.866	0.833	0.847
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	-	0.849	0.849	0.858	0.839	0.847
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering	0.850	0.850	0.856	0.843	0.849
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Leaf insertion	0.850	0.850	0.870	0.829	0.846
RST-LSTM	N-ary	$[\sigma_{\text{EDU}}, \rho_{\text{RST}}, \tau_{\text{RST}}]$	Node reordering & leaf insertion	0.848	0.848	0.851	0.845	0.848

Table 6: Predictive performance reported for the test set from the small-scale dataset with 50,000 movie reviews from the IMDb. The best values for each metric are highlighted in bold.

previous experiments). That is, we feed the networks by utilizing EDU-level features that originate from the previous dictionary-based sentiment scores. The performance measurements indicate that the resulting predictive accuracy is inferior to the RST-LSTM. For the dataset from Rotten Tomatoes, their approach achieves a balanced accuracy of 0.761 and thus represents a decline of 0.016 (i. e. -2.1%) compared to the best-performing N -ary RST-LSTM. In the case of IMDb reviews, their approach yields an accuracy of 0.831, while the N -ary RST-LSTM achieves 0.850. Hence, this work results in an improvement of 0.019 (i. e. 2.3%).

5.4. Discussion

We now investigate the trained weights of our tensor-based attention mechanism inside the RST-LSTM. This facilitates insights into how the neural network processes the discourse and infers the sentiment from the semantic structure of textual materials. Figure 9 compares the normalized weights of the tensors $U_m^{(u)}$ across different relation types m . The values result from using a child-sum RST-LSTM without data augmentation. Overall, the tensor weights between both datasets are highly correlated. The corresponding correlation coefficient stands at 0.640, statistically significant at the 99% level. However, we observe large differences in the relative importance across the relation types. For instance, relation types such as *background* and *textual-organization* entail only a marginal importance, consistent with initial expectations. In contrast, the *joint* relation yields among the highest weights across both datasets.

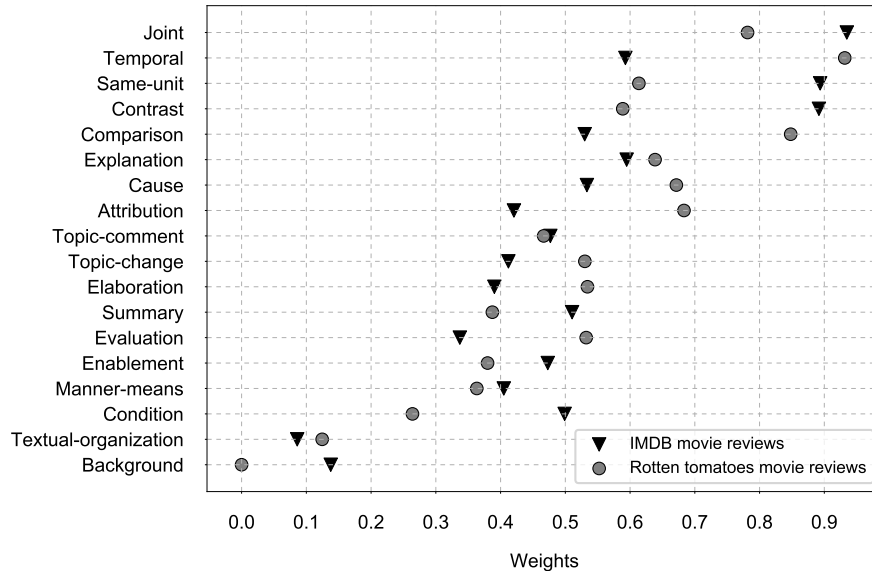


Figure 9: The weights from the attention mechanism reveal the relative importance of the discourse. More precisely, the plots shows the normalized weights of the tensor $U_m^{(u)}$ across different relation types m . The results stem from the child-sum RST-LSTM without data augmentation.

With regard to the hierarchy-related tensors, we find a greater importance (i.e. higher weights) for nuclei, as compared to satellites. For instance, the IMDB movie reviews lead to a nucleus weight of 0.738, whereas the weight of satellites totals a mere 0.588. This is in keeping with our intuition and the idea of RST since, in most cases, nuclei are more essential to the writer’s purpose than satellites.

Figure 10 illustrates the attention mechanism with an example movie review drawn from the Rotten Tomatoes dataset. Here we color the text according to the attention values inside the child-sum RST-LSTM without data augmentation. A darker text color refers to more essential pieces of information. For instance, the RST-LSTM assigns among the highest relevance to the passage “the plot is deceptively simple”, whereas “a popular-but-slow jock to run” ranges among the lowest.

every now and then a movie comes along from a suspect studio, with every indication that | it will be a stinker, | and to everybody’s surprise (perhaps even the studio) the film becomes a critical darling. | mtv films’ election , a high school comedy | starring matthew broderick and reese witherspoon, is a current example. |did anybody know this film existed a week before it opened? |the plot is deceptively simple. |george washington carver high school is having student elections. | tracy flick (reese witherspoon) is an over-achiever with her hand | raised at nearly every question , way , way , high. |mr. | ” m ” (matthew broderick), sick of the megalomaniac student, encourages paul, |a popular-but-slow jock to run. |and paul’s nihilistic sister | jumps in the race as well, for personal reasons . . .

Figure 10: Example movie review from Rotten Tomatoes after pre-processing. Individual EDUs are separated by vertical bars. A darker text color highlights more relevant passages as measured by a higher attention inside the RST-LSTM.

The above discussion confirms that the tensors build an attention mechanism that learns to weight the importance of sentences based on their position and relations in the discourse tree. As a result, the RST-LSTM can localize the relevant parts of the document and ascertain the relative importance of sentiment scores

6. Conclusion

Deep learning for natural language predominantly builds upon sequential models such as LSTMs in order to incorporate context. While these models usually achieve a high predictive power when applied to short texts, the complexity of linguistic discourse hampers performance in relation to longer documents. As a remedy, our paper proposes an innovative, discourse-aware approach: we first parse the semantic structure based on rhetorical structure theory, thereby mapping the document onto a discourse tree that encodes its storyline. We then

apply tailored tree-structured deep neural networks with an additional attention mechanism that enables us to directly learn the complete discourse tree. Each of the architectures entails more than 10,000 parameters, empowering the models to learn highly non-linear relationships.

Our findings reveal that our RST-LSTM substantially outperforms the baselines. For instance, the best-performing RST-LSTMs achieve an improvement of 4.33 % and 3.16 % in the F1-score for both datasets as compared to using word features. These gains are partially owed to our techniques for data augmentation, which slightly alter existing trees in order to enlarge the size of the training set. Evidently, data augmentation presents a viable option to reduce the risk of overfitting. Furthermore, the underlying attention mechanism learns the relative importance of passages based on their position in the discourse tree. This facilitates insights into which discourse units convey essential pieces of information. Altogether, our work contributes to research in the field of text mining by advancing deep learning methods towards including semantic information when processing natural language.

Future research could benefit from a neural network in which the dictionary-based approach is replaced by sentiment scores that are learned from the content itself. For instance, one could think of an alternating approach where, in one step, the algorithm trains the RST-LSTM and, in every second step, optimizes an autoencoder based on the content of the EDUs. This could theoretically yield additional improvements in the predictive performance.

References

- [1] B. Pang, L. Lee, Opinion mining and sentiment analysis, *Foundations and Trends in Information Retrieval* 2 (2008) 1–135.
- [2] R. Feldman, Techniques and applications for sentiment analysis, *Communications of the ACM* 56 (2013) 82–89.
- [3] K. Xu, S. S. Liao, J. Li, Y. Song, Mining comparative opinions from

- customer reviews for competitive intelligence, *Decision Support Systems* 50 (2011) 743–754.
- [4] A. Bhattacharjee, An empirical analysis of the antecedents of electronic commerce service continuance, *Decision Support Systems* 32 (2001) 201–214.
 - [5] B. Pang, L. Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, in: *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*, 2005, pp. 115–124.
 - [6] X. Yu, Y. Liu, X. Huang, A. An, Mining online reviews for predicting sales performance: A case study in the movie domain, *IEEE Transactions on Knowledge and Data Engineering* 24 (2012) 720–734.
 - [7] S. Tirunillai, G. J. Tellis, Does chatter really matter? Dynamics of user-generated content and stock performance, *Marketing Science* 31 (2012) 198–215.
 - [8] S. Feuerriegel, H. Prendinger, News-based trading strategies, *Decision Support Systems* 90 (2016) 65–74.
 - [9] N. Pröllochs, S. Feuerriegel, D. Neumann, Negation scope detection in sentiment analysis: Decision support for news-driven trading, *Decision Support Systems* 88 (2016) 67–75.
 - [10] H. Rui, Y. Liu, A. Whinston, Whose and what chatter matters? The effect of tweets on movie sales, *Decision Support Systems* 55 (2013) 863–870.
 - [11] C. D. Manning, H. Schütze, *Foundations Of Statistical Natural Language Processing*, MIT Press, Cambridge MA, 1999.
 - [12] J. Hirschberg, C. D. Manning, Advances in natural language processing, *Science* 349 (2015) 261–266.

- [13] W. C. Mann, S. A. Thompson, Rhetorical structure theory: Toward a functional theory of text organization, *Text-Interdisciplinary Journal for the Study of Discourse* 8 (1988) 243–281.
- [14] A. Hogenboom, F. Frasincar, F. de Jong, U. Kaymak, Using rhetorical structure in sentiment analysis, *Communications of the ACM* 58 (2015) 69–77.
- [15] J. Märkle-Huß, S. Feuerriegel, H. Prendinger, Improving sentiment analysis with document-level semantic relationships from rhetoric discourse structures, *50th Hawaii International Conference on System Sciences* (2017).
- [16] A. Hogenboom, F. Frasincar, F. de Jong, U. Kaymak, Polarity classification using structure-based vector representations of text, *Decision Support Systems* 74 (2015) 46–56.
- [17] Y. Ji, N. Smith, Neural discourse structure for text categorization, *arXiv preprint arXiv:1702.01829* (2017).
- [18] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Transactions on Neural Networks* 5 (1994) 157–166.
- [19] B. Heerschop, F. Goossen, A. Hogenboom, F. Frasincar, U. Kaymak, F. de Jong, Polarity analysis of texts using discourse structure, in: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*, 2011, pp. 1061–1070.
- [20] H. Hernault, H. Prendinger, D. A. DuVerle, M. Ishizuka, T. Paek, Hilda: A discourse parser using support vector machine classification, *Dialogue and Discourse* 1 (2010) 1–33.
- [21] Y. Ji, J. Eisenstein, Representation learning for text-level discourse parsing, in: *Proceedings of the 52nd Annual Meeting Annual Meeting on Association for Computational Linguistics (ACL '14)*, 2014, pp. 13–24.

- [22] C. Zirn, M. Niepert, H. Stuckenschmidt, M. Strube, Fine-grained sentiment analysis with structural features, in: Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP '11), 2011, pp. 336–344.
- [23] M. Taboada, K. Voll, J. Brooke, Extracting sentiment as a function of discourse structure and topicality, Simon Fraser University School of Computing Science Technical Report (2008).
- [24] J. M. Chenlo, A. Hogenboom, D. E. Losada, Rhetorical structure theory for polarity estimation: An experimental study, *Data & Knowledge Engineering* 94 (2014) 135–147.
- [25] P. Bhatia, Y. Ji, J. Eisenstein, Better document-level sentiment analysis from RST discourse parsing, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '15), 2015, pp. 2212–2218.
- [26] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning: Adaptive Computation And Machine Learning Series*, MIT Press, Cambridge MA, 2017.
- [27] D. Williams, G. E. Hinton, Learning representations by back-propagating errors, *Nature* 323 (1986) 533–536.
- [28] S. Hochreiter, J. Schmidhuber, Long short-term memory, *Neural Computation* 9 (1997) 1735–1780.
- [29] K. S. Tai, R. Socher, C. D. Manning, Improved semantic representations from tree-structured long short-term memory networks, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL '15), 2015, pp. 1556–1566.
- [30] S. Baccianella, A. Esuli, F. Sebastiani, SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC '10), 2010, pp. 2200–2204.

- [31] B. Pang, L. Lee, A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts, in: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04), 2004, pp. 271–278.
- [32] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th Annual Meeting on Association for Computational Linguistics (ACL '11), 2011, pp. 142–150.
- [33] M. F. Porter, An algorithm for suffix stripping, Program 14 (1980) 130–137.